Introduction to Linux

Cecina Babich Morrow

2024-02-16

Kernel: Linux, MacOs, Windows Shell: Ksh, Zsh, Bash, cmd, Powershell Like the language you're speaking to the kernel Typically interact with shell, not the kernel directly Bash is the most common (Bourne Again SHell) Programs to use the shell: Gnome Terminal, Konsole, MacOs Terminal, Git Bash, Powershell The thing you actually open to interact with the shell

Working with files

`ls`

`ls` is a command-line utility for listing the contents of a directory. Here are some common flags used with `ls`:

- `ls -l` gives you a long list of files in the directory, including the permissions, the number of links, the owner, the group, the size, the date, and the name of the file
- `ls -lt` gives you a long list of files in the directory, sorted by time
- `ls -ltr` gives you a long list of files in the directory, sorted by time in reverse order `ls -hl` gives you the file sizes in human-readable format

File permissions

When running `ls -l`, we can see the permissions of each file. For instance, if the permissions are `drwxr-xr-x`, this is how to interpret that:

- `d` means it is a directory
- 3 triples: what is the owner allowed to do, what is the group allowed to do, and what is everyone else allowed to do
- `rwx` means that someone can read, write, and execute
- Anything replaced with `-` is an action that that entity cannot do, e.g. `r-x` means that the entity can read and execute, but not write

Some exercises

Can you mv one file to more than one destination? (Couldn't do this)

What happens if you give cp no destination?

```
cp data2
```

The result is:

```
cp: missing destination file operand after 'data2'
Try 'cp --help' for more information.
```

Can you cp, but without being inside the folder it is located in?

```
cp folder_copy/file2 folder_copy_file2
```

Can you move a folder?

```
mv folder_copy new_folder_copy
```

How does moving a file onto a folder behave?

```
mv data2 new_folder_copy
cd new_folder_copy/
ls
```

```
## mv: cannot stat 'data2': No such file or directory
## bash: line 2: cd: new_folder_copy/: No such file or directory
## 01_intro_to_c++.pdf
## 01_intro_to_c++.Rmd
## 02_interfacing_r_with_c++
## 03_advanced_rcpp_1
## 04_intro_to_linux.Rmd
```

The results are:

```
copy-of-data2-but-HERE  data2  file1  file2  file3
```

Can you mv the folder you are actually in?

```
mv . new_folder_copy2
```

You get `mv: cannot move '.' to 'new_folder_copy2': Device or resource busy`.

Viewing the contents of files

Some options for viewing the contents of files:

- `cat`: prints the contents of a file to the terminal
- `head` & `tail`: prints the first or last 10 lines of a file, respectively (you can change the number of lines printed by running `head -n 5 file.txt`, for example)
- `tac`: prints the contents of a file in reverse by line (i.e. the last line is printed first)
- `rev`: prints the contents of a file with the contents of each line reversed by character
- `uniq`: prints the contents of a file with adjacent identical lines collapsed to one
- `sort`: prints the contents of a file sorted
- `less`: allows you to scroll through the contents of a file. If you want to find a specific string, you can type `/` and then the string you want to find, and then press `n` to find the next instance of that string

Editing files

Some popular text editors:

- `nano`: a simple text editor
- `micro`: a modern version of `nano`
- `emacs`: a powerful text editor with higher learning curve than `nano` / `micro`
- `jove`: a more minimal version of `emacs`
- `vim`: a powerful text editor with a steep learning curve that uses lots of keyboard shortcuts

Searching

`grep` is a command-line utility for searching plain-text data sets for lines that match a regular expression. Here are some common flags used by `grep`:

- `grep -c`: prints the number of lines that match the pattern
  - If you have multiple targets, it will print the number of matches for each target, e.g. `grep -c "villain" king-lear.txt julius-caesar.txt` yields

```
king-lear.txt:20
julius-caesar.txt:4
```

- `grep -i`: makes the search case-insensitive
  - Combining `grep -ci` will give you the number of matches case-insensitive
- `grep -n`: prints the line number of each matching line
- `grep -cih`: prints the number of matches case-insensitive and suppresses the file names
- `grep -A2`: prints the line matching the pattern and the two lines following it
- `grep -B2`: prints the line matching the pattern and the two lines preceding it
- `grep -C2`: prints the line matching the pattern and the two lines preceding and following it (alternatively, you could run `grep -A2 -B2`)
- `grep -l`: prints the names of files with matching lines

You can use `grep` to search recursively through directories by using the `-r` flag. You can also use the wildcard `*` to search within files that match a particular pattern, e.g. `grep sparrow a*` will search for the word "sparrow" in all files that start with "a".

Searching exercises

To determine which plays contain the word "squirrel", we can run:

```
grep -l "squirrel" *
```

```
a-midsummer-nights-dream.txt
romeo-and-juliet.txt
```

Similarly, we can list the plays containing "toasted cheese":

```
grep -l "toasted cheese" *
```

```
henry-vi-part-2.txt
king-lear.txt
merry-wives-of-windsor.txt
```

Now we want to count the number of plays containing the word “confidence”:

```
grep -l "confidence" * | wc -l
```

We get the result `12`.

To get the line number of the word “folly” in Hamlet:

```
grep -n "folly" hamlet.txt
```

```
4761:   But that this folly douts it.
```

We want to search for the word “asleep” in plays that have the word “and” in their title:

```
grep -lc "asleep" *and*
```

```
antony-and-cleopatra.txt
romeo-and-juliet.txt
titus-andronicus.txt
```